



技術的負債の再利用を考慮 したソフトウェアのコスト 見積もりモデルの構築

オープンソースソフトウェア工学研究室 M1

60200135

吉富 楓雅

背景

- 近年のソフトウェアシステムは定期的に変更を加える必要がある
 - * 長期間(10年～30年)運用・保守されることを前提として開発される
 - * 顧客の要求や外部環境の変化に応じて適宜対応する必要がある
- ソフトウェアシステムに加えた変更が保守性を低下させる原因となる場合が多い
 - * 短期的には効果的であるが、長期的には保守コストが高くなる場合がある(**技術的負債**[1])

技術的負債の種類[2]

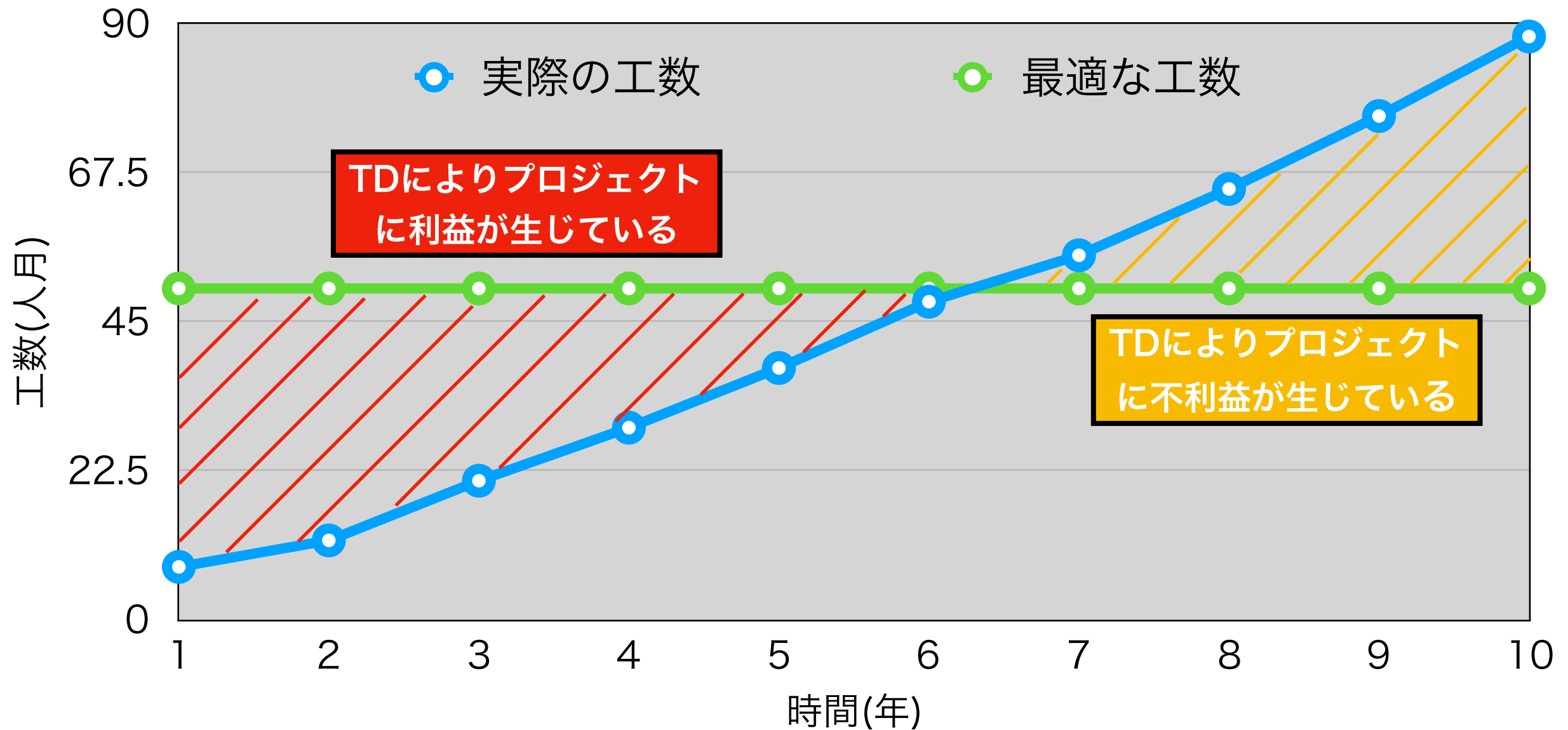
技術的負債の種類	概要
Requirements TD	最適な要求仕様と実際に実装されたシステムとの差
Architectural TD	保守性など内部品質を犠牲にしたアーキテクチャ
Design TD	詳細設計で行われる最適とは言えないショートカット
Code TD	コーディング規則に違反するような複雑なコード
Test TD	テストの欠如などテストにおける最適とは言えないショートカット
Build TD	ビルドを複雑かつ困難にするソフトウェアシステムやビルドシステム
Documentation TD	コメントの欠如などによる不十分さ、不完全さをもつドキュメント
Infrastructure TD	最適ではない開発関連のプロセス、技術、サポートツール
Versioning TD	ソースコードのバージョン管理で発生する問題
Defect TD	ソフトウェアシステムで発見されたバグ

技術的負債の研究対象[2]

技術的負債の研究対象	目的
TD identification	静的解析ツールなどを用いてシステム内のTDの検出
TD measurement	見積もり手法を用いたシステムの既知のTDの利子とコストの定量化
TD prioritization	事前に決めたルールによる, TDの優先順位の決定
TD prevention	潜在的なTDの混入防止
TD monitoring	未解決のTDのコストと利点の時間的変化の観察
TD repayment	リエンジニアリングやリファクタリングによる システム内のTDの除去, 軽減
TD representation / documentation	特定のステークホルダの懸念を解消するためのTDの体系化
TD communication	議論, 管理することを目的としたTDのステークホルダの可視化

先行研究[3]

- 技術的負債を定量化するためのアプローチ

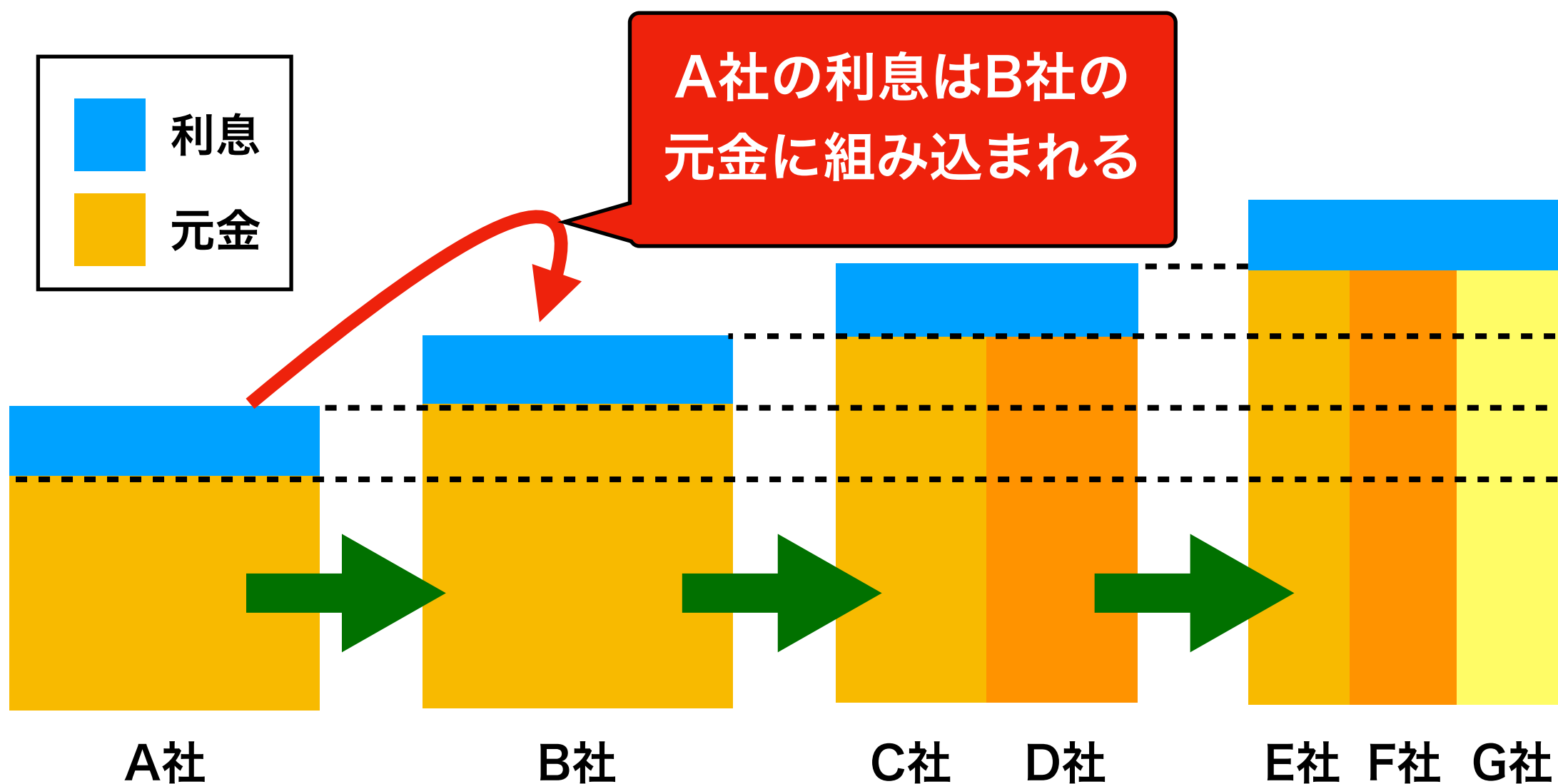


技術的負債 = 実際のメンテナンス工数 - 最適なメンテナンス工数

多重債務のメカニズム

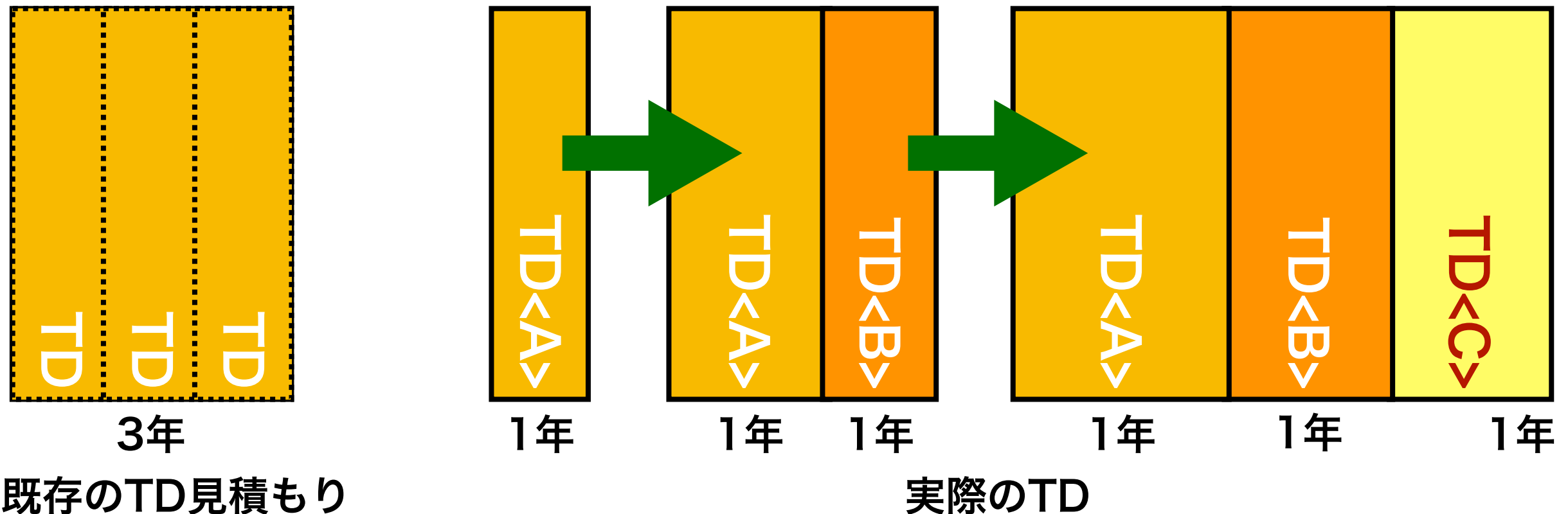
- 目の前の借金を返せず、他から借り入れた借金で返済する状態

* 借金額は雪だるま式に増える



先行研究の問題点

- 技術的負債 = 実際のメンテナンス工数 - 最適なメンテナンス工数
 - ▶ 以前の技術的負債によって技術的負債の割増が発生している可能性がある(**技術的負債の成長**)



先行研究のモデルでは無視されている

研究の目的

- 技術的負債の成長を考慮して、技術的負債工数見積もりモデルを構築すること
 - * 多重債務のメカニズムを導入する

データセット

プロジェクト名	言語	対象期間
<u>RxJava</u>	Java	2014年03月12日 ~ 2016年09月30日
<u>Jabref</u>	Java	2012年03月18日 ~ 2017年03月29日

FindBugs

- Javaプロジェクトを対象とする静的解析ツール
- FindBugsの解析結果に格納される情報
 - * **カテゴリ**：エラーを8種類に分類している
 - * **信頼度**：検出されたバグが実際にバグになりうる可能性を表している
 - * **バグの危険度**：そのバグがどの程度深刻なのかを表している
 - * **ファイルパス**：どのファイルでバグを検知したのか示されている
 - * **行数**：何行目から何行目でバグを検知したのか示されている

FindBugsのエラーカテゴリ

カテゴリ名	内容
<u>CORRECTNESS</u>	明らかなコーディングミスによる，開発者の意図しないコード
<u>SECURITY</u>	リモートから悪用可能なセキュリティ上の脆弱性につながる，信頼されていない入力方法
<u>BAD PRACTICE</u>	推奨される，あるいは本質的なコーディング規範からの逸脱
<u>STYLE</u>	誤解を与えやすいコード，変則的，間違いを招きやすいコード
<u>PERFORMANCE</u>	必ずしも間違いではないが，実行効率が悪くなる可能性のあるもの
<u>MALICIOUS CODE</u>	信頼されていないコードからの攻撃に対して脆弱性を持つコード
<u>MT CORRECTNESS</u>	スレッド・ロックの扱いに対するコードの欠陥
<u>I18N</u>	国際化・ローカルの扱いに対するコードの欠陥

調査項目

1. TDにはどのような種類があるか

- * エラーカテゴリ毎に分布をとり，修正されずに残り続けるTDを調査する

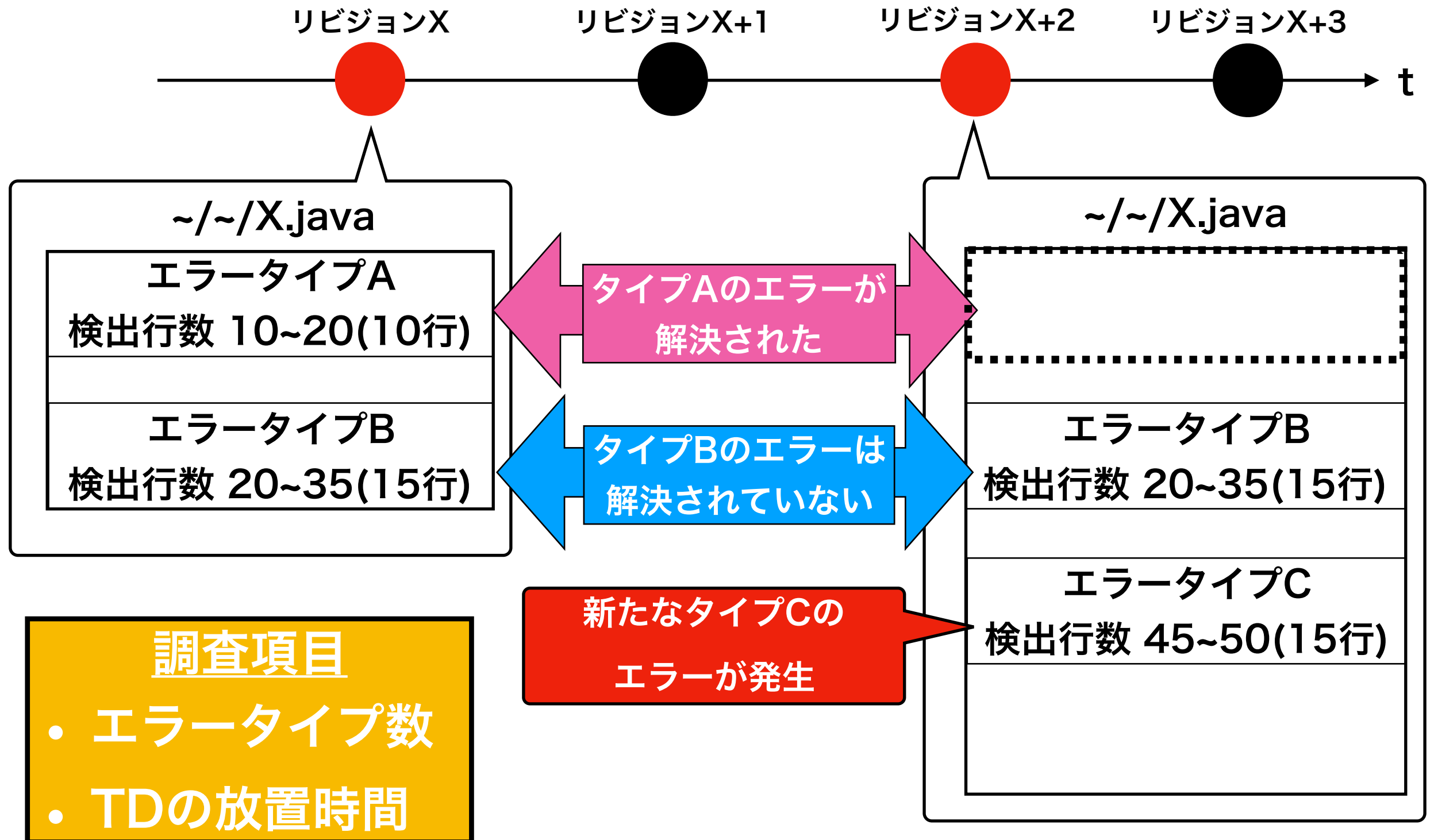
2. TDを含むファイルは再利用されるのか

- * TDを含むファイルがいくつ呼び出されているのか調査する

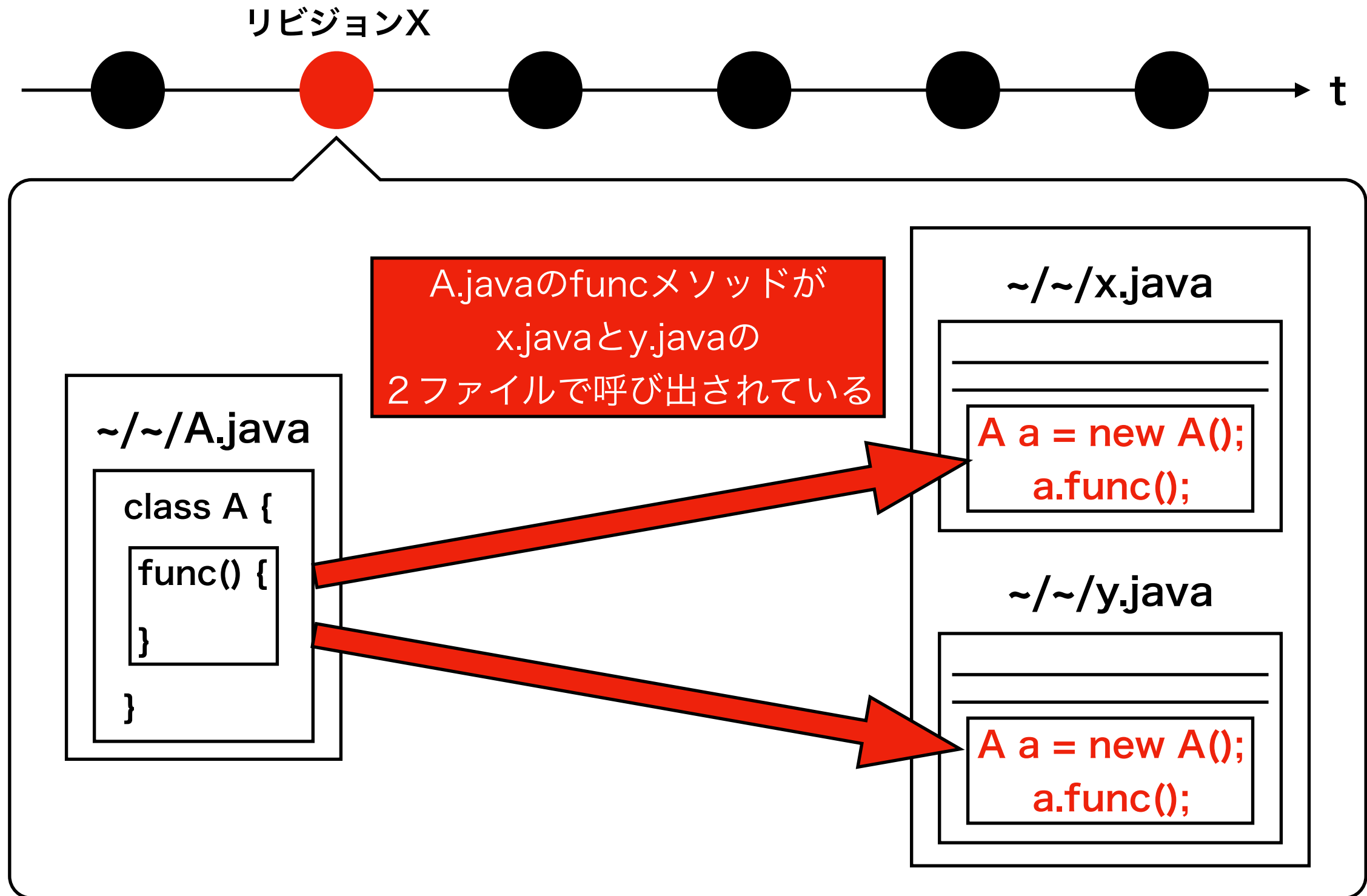
3. TDはどのようにして解決されるのか

- * メトリクス解析を行うことでTDを含むファイルの変更前後の差分を調査する

TDの種類のアナリ方法

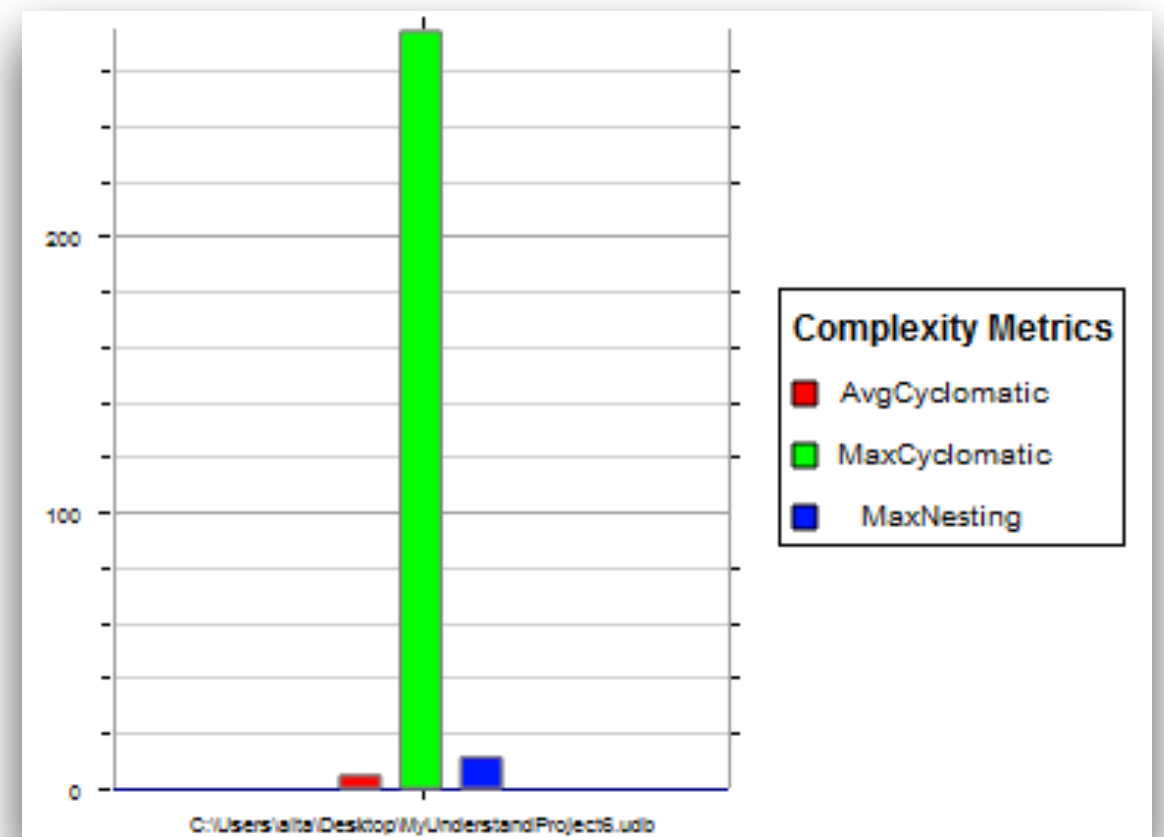


再利用されたTDを含むファイルの分析方法



ファイルのメトリクス解析

- 使用するツール：Understand
 - * 約100種類のメトリクス分析を行うことが可能
- 解析内容
 - * クラスの平均行数
 - * クラスの平均複雑度
 - * クラスの最大複雑度



エラーの修正前後でどのように変更されたのかを調査する

調査項目 1 : 分析結果

- ホゲホゲ
 - * テンプレ
 - ▶ テンプレ
- テンプレ
 - * テンプレ
- テンプレ
 - * テンプレ

調査項目 2 : 分析結果

- ホゲホゲ
 - * テンプレ
 - ▶ テンプレ
- テンプレ
 - * テンプレ
- テンプレ
 - * テンプレ

調査項目 3 : 分析結果

- ホゲホゲ
 - * テンプレ
 - ▶ テンプレ
- テンプレ
 - * テンプレ
- テンプレ
 - * テンプレ

今後の予定

- テンプレ
 - * テンプレ
 - ▶ テンプレ
- テンプレ
 - * テンプレ
- テンプレ
 - * テンプレ

テンプレ

- テンプレ
 - * テンプレ
 - ▶ テンプレ
- テンプレ
 - * テンプレ
- テンプレ
 - * テンプレ

補足資料

テンプレ

- テンプレ
 - * テンプレ
 - ▶ テンプレ
- テンプレ
 - * テンプレ
- テンプレ
 - * テンプレ

テンプレ

- テンプレ
 - * テンプレ
 - ▶ テンプレ
- テンプレ
 - * テンプレ
- テンプレ
 - * テンプレ